Universidad Autónoma de Madrid

Escuela politécnica superior

Doble Grado en Ingeniería Informática y Matemáticas

Trabajo Fin de Grado

ESTUDIO DEL TRÁFICO WEB BASADO EN LA HUELLA DNS

Autor: Miguel Rodríguez Gutiérrez

Tutor: Francisco Javier Ramos de Santiago

Ponente: Javier Aracil Rico

Mayo 2016

Universidad Autónoma de Madrid

Escuela politécnica superior

Double Degree in Mathematics and Computer Engineering

Degree Final Project

WEB TRAFFIC ANALYSIS BASED ON DNS FOOTPRINTS

Author: Miguel Rodríguez Gutiérrez

Tutor: Francisco Javier Ramos de Santiago

Secondary tutor: Javier Aracil Rico

May 2016

WEB TRAFFIC ANALYSIS BASED ON DNS FOOTPRINTS

Author: Miguel Rodríguez Gutiérrez
Tutor: Francisco Javier Ramos de Santiago
Secondary tutor: Javier Aracil Rico

High-Performance Computing and Networking Research Group
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
May 2016

Resumen

Resumen

DNS es el protocolo de red que asocia los nombres de dominio con sus correspondientes direcciones IP entre otros. La mayoría de las veces que un usuario accede a una página web utilizando un navegador se realizan múltiples peticiones DNS, dando a los operadores y otras entidades con acceso al tráfico DNS información valiosa acerca del comportamiento y hábitos de los usuarios cuando éstos navegan por la web. La monetización de las grandes cantidades de datos que los operadores tienen de sus usuarios se encuentra aún en una fase temprana. Específicamente, conocer las páginas web que los usuarios o grupos de usuarios visitan aplicando el filtrado correspondiente abre nuevas oportunidades de negocio. Sin embargo, la construcción de perfiles web de usuarios precisos en redes de gran tamaño supone un gran desafío no solo por los requisitos que conlleva capturar tráfico, sino también por el uso de cachés DNS, la proliferación de redes de bots y la complejidad de las webs actuales (cuando un usuario visita una web, se descargan automáticamente contenidos adicionales como anuncios y servicios de la misma compañía o de terceros). De este modo, proponemos contabilizar las visitas a las páginas web que realizan los usuarios por medio de las huellas DNS. Este enfoque novedoso consiste en considerar que una web fue efectivamente visitada si se encuentran tanto las peticiones DNS del dominio principal como aquellas que suceden automáticamente al realizar la visita. Este enfoque se ha codificado finalmente en una aplicación denominada DNSprints. Tras su correspondiente parametrización (equilibrar la importancia de cada nombre de dominio en una huella respecto a las demás), hemos concluido que la aplicación propuesta es capaz de identificar visitas con tasas de falsos y verdaderos positivos del 7% y el 88% respectivamente a una tasa de 500,000 paquetes por segundo, lo que prueba su aplicabilidad.

Palabras Clave

Análisis de navegación; huella DNS; perfiles web de usuario; DNSprints.

Abstract

DNS is the protocol that associates domain names to their corresponding IP addresses among others. Most of the times a user accesses a website using a web browser several DNS queries trigger, giving ISPs and other entities that have access to DNS traffic valuable information about users' browsing behavior and habits. The monetization of the large amount of data that ISPs have of their users is still in early stages. Specifically, the knowledge of the websites that users or groups of users visit opens new opportunities of business, after the convenient sanitization. However, the construction of accurate web-user profiles on large networks is a challenge not only because the requirements that capturing traffic entails, but also given the use of DNS caches, the proliferation of botnets and the complexity of current websites (i.e., when a user visit a web a set of self-triggered websites, such as banners and both same company and third parties services are downloaded). In this way, we propose to count the visits users make to websites by means of DNS footprints. Such novel approach consists of considering that a website was actively visited if the DNS queries of both the root domain and the set of self-triggered websites are found. This approach has been coded in final application named DNSprints. After its parametrization (i.e., balancing the importance of each domain name in a footprint with respect to the total set of footprints), we have measured that our proposal is able to identify active visits with false and true positives rates below 7% and over 88% respectively achieving rates of 500,000 packets per second, thus proving its applicability.

Key words

Browsing analytics; DNS footprint; web-user profiles; DNSprints.

Acknowledgments

I would like to thank *High Performance Computing and Networking* research group members for their extremely valuable and constant help, support and feedback. Specially to my tutors Javier Ramos and Javier Aracil, and to Jose Luis García-Dorado and Germán Retamosa for working hand in hand with me to make the best out of this project.

Thanks as well to my university teachers for taking care of their students' learning process and sharing their knowledge and expertise with dedication year after year.

Finally, I would also like to thank all my college classmates and friends for their disinterested dedication and hard work to help each other out and all the enjoyable moments spent together. It has been a great pleasure to share these years with you.

Contents

List of Figures

1

Introduction

This project, which has been developed in collaboration with the *High-Performance Computing and Networking* research group, has the purpose of designing and implementing an analysis tool that can track the websites visited by users based exclusively on the DNS traffic. This traffic analysis and monitoring tool has been named DNSprints.

1.1 Motivation

DNS is the well-known protocol used for resolving domain names into IP addresses. Every time a user accesses a website, its domain name has to be translated into an IP address using DNS protocol before the user can actually access to the website's content. Therefore analyzing DNS traffic can provide useful insight on how users behave while browsing the Internet.

The commercial exploitation of web browsing analytics is currently a fact, whereby the collaboration between marketing and Internet communities has progressively strengthened after each business success. The information about users' preferences that the different actors in the Internet arena recollect can be used not only as inputs for tailored online-ads but for further and novel mechanisms to monetize the rich set of available data [1, 2], even more with the advent of the Big Data era. In paying attention to web access profiles, it becomes apparent that the identification of a competitor attracting the interest among the customers of a given company is of paramount interest to react accordingly. Similarly, the interest in terms of visits of the population of a given geographical area is a useful input to plan the expansion for many retailers. A real-world example of this is Google AdWords service [3], such that Google sells

keywords, statistics of keywords, and search results of its users [4] and customizes ads according to users' profiles.

Nonetheless, the capacity of creating users' web-profiles is not the exclusive preserve of web search engines as of today, but also others, such as ISPs, have the opportunities to exploit these business models. Actually, in a richer fashion given that search engines are only aware of the few first accesses users make to a web site. After that, typically, web names are retained in browser recent history (or bookmarks), or simple memorized. This way, search engines are no longer used as gateways to such sites.

Alternatively to client intrusive software and add-ons installed on clients' web browsers, such as toolbars [5] or cookies [6], the first ISP approach to this issue is the apparently simple task of capturing and inspecting HTTP traffic [7]. However, two concerns arise, the unstoppable increase of HTTPS and the implicit difficulty of capturing large volumes of HTTP traffic (needless to say inspecting it) potentially distributed in different geographical areas. Facing the former, the authors in [8, 9] pointed out the role that DNS could play to reveal the traffic behind a HTTPS flow. Further, the authors in [10, 11] successfully correlated traffic flows [12] and temporally-close DNS queries, so identifying IP addresses of users and the name of the accessed hosts. Unfortunately, such approaches still require to collect all HTTP and DNS traffic to construct the flows to which relate the DNS protocol, and, importantly, other collateral issues emerged:

- DNS Cache: Most of DNS resolvers and clients implement a cache for DNS traffic where the associated IP of a given domain name is temporarily stored. As Time To Live (TTL) for the cache entry can be long [11], a point of presence monitoring traffic potentially cannot see clients' DNS requests although they are effectively visiting a web.
- Automated clients and botnets [13]: Often, DNS queries are not generated by actual users from a web browser, but from scripts/bots that poll DNS servers searching for updates or information about the service.
- The tangled web [10]: Almost any single web includes a set of external contents (e.g., resources of other webs of the same company or, simply, banners or ads) whose domain names have to be resolved for fully loading the requested web itself. This causes that additional DNS traffic is generated without being explicitly requested by users. Such self-triggered traffic makes users' web-profiles imprecise especially in business-terms.

1.2 The DNS footprint

The DNS footprint is the key concept in which DNSprints is based. The DNS footprint's purpose is to exploit the tangled characteristic of websites. As introduced before, fully loading a website on a browser does not only trigger a single DNS request to the domain typed in the browser. Loading the website contents and the external services cause many DNS resolutions to trigger.

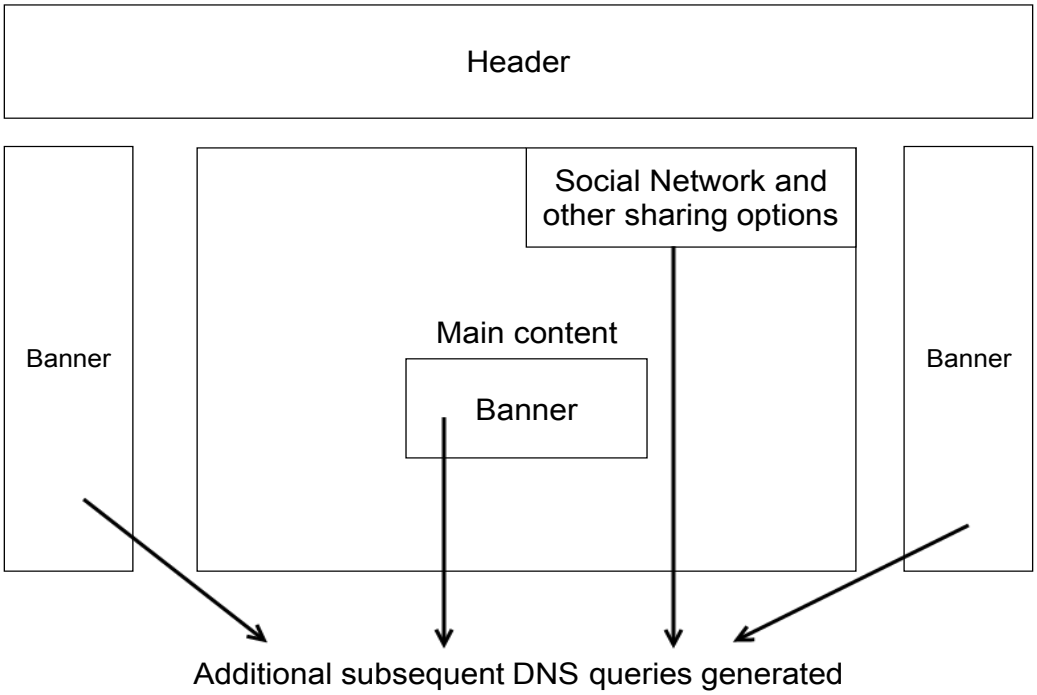

Figure 1.1: Sample website layout, with content that generates self-triggered DNS queries embedded on several banners and external services

This set of domains directly or indirectly resolved when a website is visited using a web browser conform the DNS footprint of that website. It is important to be aware that the DNS footprint of a website depends on the device, browser and OS used and the time when the query was made. DNSprints tackles this issue by intersecting different footprints and defining partial and final footprints, as explained in the upcoming sections.

Figure 1.1 features a sample layout for a website with several embedded ads, banners and links to both external and same company services. In this way, after the main website is requested, while users wait for loading, a set of DNS queries/responses are generated.

Such behavior is illustrated in Figure 1.2, where multiple resolution processes are in place. The set of different resolved domains that a given website triggers defines the DNS footprint of such a website. This way, given a list of root-domain websites, i.e., those websites of interest for business purposes, we construct their DNS footprints that will be used to search in traffic aggregates to determinate the visits per user, potentially sanitized, or per aggregation in market segments and geographical areas.

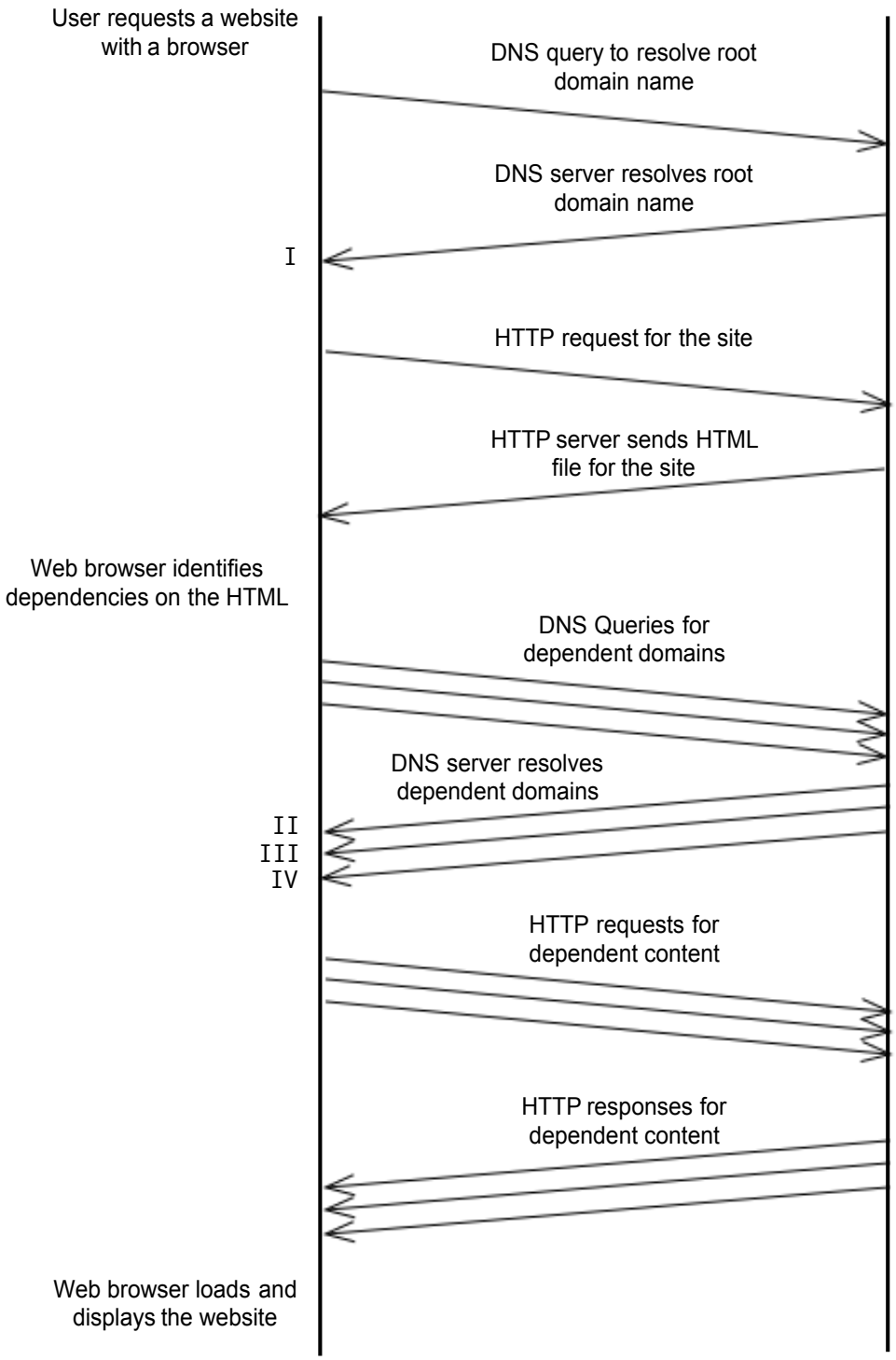

Figure 1.2: Flow chart describing the packet flow of DNS and HTTP traffic when a site is requested using a web browser

1.3 DNSprints features

In this light, we search a UDP traffic inspection mechanism to detect every time a user accesses a website circumventing the problem that some packets are hidden by caches, and, importantly, with the capacity of distinguishing between a requested web and other self-triggered web traffic.

DNSprints' purpose is to analyze the traffic of a large DNS server. Having this in mind, DNSprints has been designed to run on real-time, high-throughput environments, and performance analysis has been performed.

In fact, being less restrictive with the footprint (i.e., accept a match although not all the websites of a footprint are found) circumvents the cache problem while being more restrictive will rule out self-triggered visits, in other words "false" resolutions, with respect to the actual web requested by the user. How to parametrize this is studied along this work. In addition, we remark that not all websites that take part in a footprint must be considered at same level. If a website appears regularly in many footprints, its capacity of discrimination is limited, and the opposite for uncommon website. Similarly, by the time footprints are constructed some pairs of websites appear with high probability together. Consequently, we propose to weight the importance of a website by its number of occurrences in the full set of constructed footprints, and also by the times it appears in the training period for footprints construction.

We have applied it in traces captured in our laboratory including a significant set of root-domain websites as well as diverse cross traffic. The results show a performance of 500,000 packets per second and a false and true positives rates below 7% and over 88% respectively, thus proving the benefits and accuracy of DNSprints.

1.4 Document structure

The following chapter exposes the state of the art, DNS-based monitoring techniques and some previous works are analyzed pointing out the main difference with our proposals. Chapter 3 explains the DNS detection method used by DNSprints tool. Next the architecture and specific implementation issues are described in Chapter 4, followed by the results obtained from the application of the methodology to real traffic traces, which are detailed in Chapter 5. Finally, some conclusions and future work lines are given in Chapter 6.

2

State of the art

2.1 Common monitoring techniques

2.1.1 Deep Packet Inspection (DPI)

DPI is a monitoring technique based on full analysis of the flows and payload of the packets. Therefore, it requires complete access to the traffic of the network being monitored and knowledge of the protocols that are being used. Using DPI in the scenario proposed in this project is not possible, since only access to DNS traffic is granted.

2.1.2 Standard DNS analysis

We also consider a standard DNS analysis method. This method considers a visit to a website every time its domain name is resolved. As it was introduced before, this has some counterparts compared to DNSprints:

- Standard method is not aware of DNS cache. This way, its accuracy is drastically reduced when monitoring websites with large DNS TTL on their domain, as that domain will not be resolved again even if the user accesses it multiple times.
- Standard method counts DNS resolutions caused by pings, bots or web crawlers as user visits. Every interaction with a website triggers a DNS query for its domain name, this way, events such as pings that only resolve the root domain of the website will be counted

as visits. On the other hand, DNSprints will not consider this events as visits to the site, since they will not trigger resolutions for the DNS footprint of the website.

2.2 Related work

The typical mechanisms for measuring the relative popularity of websites were based on intrusive software and tracker addons installed on clients' web browsers such as toolbars. Between these proposals, Alexa [5] stands out although the number of users is also remarkable in ComScore [14] and NetRatings [15]. Unfortunately, the information this approach provides is usually too aggregated and, importantly, is only based on volunteers' traffic, what does not guarantee a representative population. Alternatively, Google with its AdWords service [3] analyses the queries users carry out on its interface and exploits this information commercially. This approach is limited by the fact that users tend to search for website only the first times they visit it, after this, tools such as recent-history suggestions or bookmarks are used for the subsequent accesses to those websites. In addition, our focus is on the opportunities that ISPs have in this area, instead of search-engine companies' ones as they in fact are monetizing their queries since years ago.

The natural approach from ISPs is to sniff HTTP traffic [16, 7] and interpret it. However, this approach comes together with limitations such as the proliferation of traffic encryption and the difficulty of storing, capturing and analyzing traces (and especially, HTTP traffic given its relevance in volume in the current Internet). The answer from the research community was to turn to DNS protocol. DNS traffic analysis has been for long a topic for research, and its performance, security and behavior have been extensively studied. There are several tools for displaying and analyzing DNS traffic such as TreeTop [8] or dnstop [17]. However, none of them provides an accurate method for relating web traffic and user behavior.

In this regard, we remark several efforts to link ephemeral IP addresses and users apart from entrusting RADIUS records with this task. Specifically, Herrmann et al. [18] developed several DNS traffic analysis techniques for tracking users based on their daily behavior. In particular, in using the previous day traffic records, it was possible to re-identify the users the day after by recognizing their browsing habits on a daily-changing dynamic IP addressing environment. We note that the problem we face is almost the opposite. Instead that the users' habits permit us to tracker them, we pursue to identify user behavior to create web-profiles.

Rajab et al. [19] related the probability of a host name to be in a DNS server's cache to the way users' access to websites. The result is an estimation of the density of users accessing to certain domains. As positive points, this method does not need access to the DNS traffic (but periodically polling DNS servers, although many if these servers are not vulnerable to such cache snooping) and the users privacy is observed. However, the grain and precision of the information that this mechanism provides is not enough for the most of the commercial analysis we introduced before.

This way, the research community turned its attention to tag flow records according to the DNS answers triggered by such flows. This approach followed by Plonka et al. [8, 9] and Bermudez et al. [10] output a set of flows whose destination addresses is labeled by the host name servers derived from the DNS. This approach was further improved by Mori et al. [11] who payed especial attention to the local cache problem when a DNS query is triggered by a final client. Essentially, much DNS traffic was missing due to such local caches which impeded flows from being labeled at all. They introduced a domain name graph representation whereby the relation between IPs and real host names was stored and shared between different queries over time. So, previous resolved queries were used as a replacement for other queries missing.

With respect to these works, DNSprints only needs access to the DNS traffic instead of HTTP/HTTPS, which significantly simplifies and cheapens the regular operation, especially at large and distributed networks. Moreover, another important distinguishing characteristic, DNSprints is robust to self-triggered traffic, as well as to bots and automatics DNS queries. We remark that this is key for constructing accurate and marketable web-profiles, otherwise such profiles would lead to erroneous conclusions, e.g., a user is interested in some service when it is not true. Finally, most of the previous works are sensitive to local caches, however DNSprints counts a user visit when a significant fraction of the DNS responses that a root domain triggers, i.e., a footprint, are resolved. Such flexibility implies that the loss of some responses only affects marginally.

3

Formal description

This chapter describes the method used by DNSprints to identify user requests based on DNS queries. Our purpose is to detect when a user has requested a certain website using a web browser only by inspecting the DNS resolutions generated.

Let $D = \{d_1, d_2 \dots d_N\}$ be the set of websites to be monitored. As we stated before, when a user requests a website d_i, several resolutions are self-triggered in addition to the domain name explicitly requested. All these resolved domain names make up a set of subdomains that define the DNS footprint F_i for that website, being $F_i = \{d_i^1, d_i^2 \dots d_i^{M_i}\}$, $i = 1, 2, \dots N$. After a domain name d_i^j has been resolved, clients usually store that information on a DNS cache for a certain time T suggested by the DNS server. The time each domain is stored on the DNS cache, called TTL, varies significantly depending on the domain. Typically, the root domain presents a longer TTL and many of the subsequent resolutions have shorter ones (e.g., <30 seconds).

Figure 3.1 shows the different values that T takes for the subdomains requested while visiting one important news website, which characterize its footprint. Such values present a very wide range, from a few seconds to several hours, with the root domain DNS cache entry lasting for more than an hour. This illustrates the importance of being aware of DNS cache. That is, if a user reads the news every few minutes, a standard DNS monitoring tool would only count one request to the root domain every several hours. With our method, the accuracy is not only dependent on the value that T announces for the root domain but also to the lowest TTL value in all the footprint F_i. This allows us to improve the detection resolution from several hours to a few minutes or even seconds in the best case.

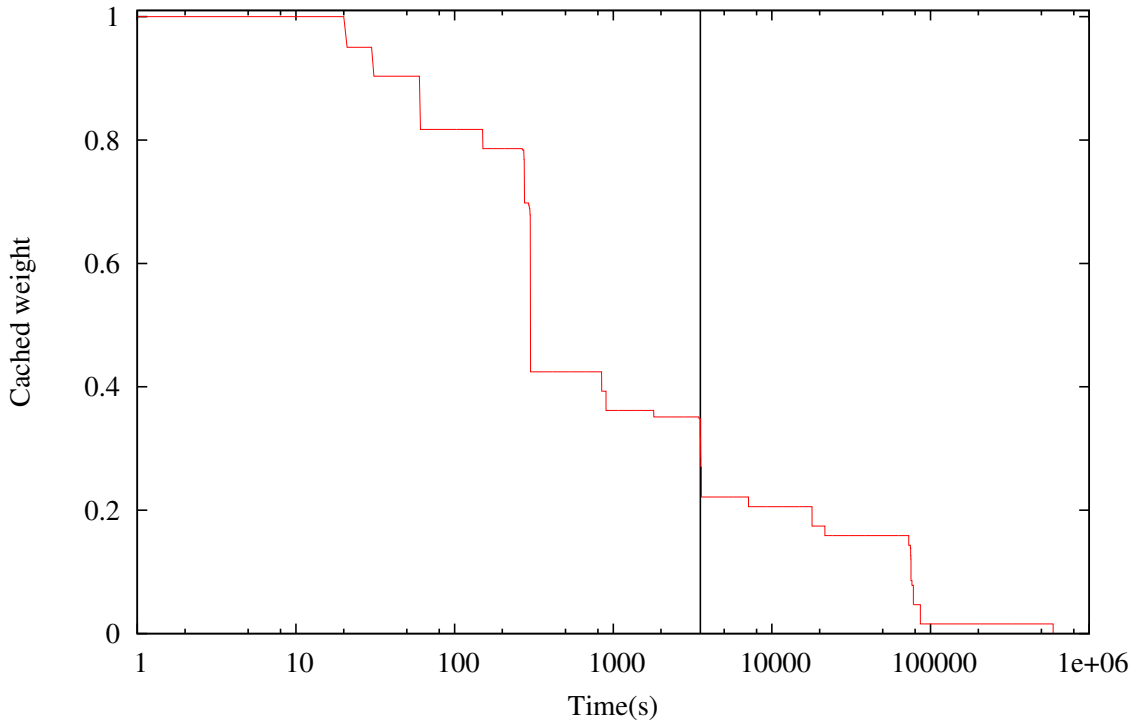

Figure 3.1: Different TTLs of the footprint domains decreasing with time. Note each domain in the footprint has a different weight. The vertical line represents the TTL of the root domain

For every domain name d_i^j present in a footprint a weight w_i^j is assigned. Such weight is inversely proportional to the number of footprints that contain that such domain name and then normalized in a way that the sum of all weights in a footprint is equal to 1. The following equations shows how the weight is calculated:

$$\tilde{w}_i^j = \frac{1}{\left|\{x : d_i^j \in F_x\}\right|} \qquad w_i^j = \frac{\tilde{w}_i^j}{\sum_{k=1}^{N} \tilde{w}_i^k}$$

The inversely proportional weighting process is performed to give more importance to the most relevant entries in the footprint and depreciate the importance of domains present in multiple footprints. For example, www.google.com or www.facebook.com will be present in the DNS footprint of many websites due to ads or social media banners. Resolving www.google.com does not provide much information about which website is being visited and, therefore, its weight should be low. On the other hand, resolving www.cnn.com is not so common and it will probably appear in fewer footprints, hence giving much more information than the former and presenting a higher weight.

If the host or browser has a DNS cache, not all the domains from F_i will be resolved every time the domain d_i is visited. In this light, domains with a low value for T will be the most frequent to be resolved as they expire soon from the cache and need to be updated regularly.

Namely, whenever a DNS server receives a query for a domain d_i^j, it is likely that will not be queried again by this user until its DNS cache entry expires.

To determine whether a user is browsing a certain monitored website d_i at a certain moment, all the weights w_i^j of the domains in F_i that are currently in this user's cache (i.e., the elapsed time is less than T) are summed. Since a request for a website should resolve all its footprint domains $d_i^1, \dots d_i^{M_i}$, it is expected that they will be cached for some time after the website is visited. The higher the current sum of the weights the most likely it is that the user is currently browsing such a website. When the user stops browsing the website, the corresponding DNS cache entries will start expiring and therefore the amount of domains cached and the sum of their weights will drop. Using this mechanism we can define a time series S_i for each website d_i and user that measures the likelihood of that user being currently visiting the website d_i. S_i is calculated as shown in the following equation:

$$S_i = \sum_{d_i^j \ is \ cached} w_i^j$$

Note that the time series S_i still reflects the likelihood of a website being visited even if the DNS caching is not present. As S_i is computed based on the value of T provided by the DNS server, which is independent of the actual content of the user cache, it is impossible to know if a host or browser is caching the names that the DNS server provides for the time it suggests.

However, the DNS cache entries do not always remain exactly for as long the time T indicates. There are several clients that do not implement a DNS caching mechanism or violate the times indicated by the DNS server [20].

Regarding such behavior, we define a certain threshold α that S_i has to exceed in order to mark a website as visited. Whenever S_i surpasses the given threshold α, a visit for that website is counted. We consider the finalization of the visit when S_i drops below α. As expected, the accuracy of the method is directly related to the value of α. A low value for α will increase the number of false positives (a website is considered as visited but the user has not requested it), while reducing the time resolution of the method. Whenever a web is visited, it is necessary to wait until the current weights sum of such website's footprint drops below α. As a consequence, all subsequent visits in a short period of time will not be considered. On the other hand, a higher value for α improves the time accuracy but also the probability of missing an actual web request from a user.

Let us illustrate the operation with examples. Figure 3.2 shows an example of the footprints F_i of N root domains along with the weight of each subdomain and TTL. The figure is represented in a matrix form where each column corresponds to a root domain and the rows correspond to the union subset of subdomains. That is, all the different subdomains observed after resolving the root domains d_1 to d_N.

Figure 3.3 illustrates the temporal evolution in the detection of visits to the website corre-

$$
\begin{array}{c|cccc|c}
 & d_1 & d_2 & \dots & d_N & \text{TTL} \\
d_1^1 & 0.35 & \dots & \dots & \dots & 178 \\
d_1^2 & 0.15 & \dots & \dots & \dots & 60 \\
d_1^3 & 0.2 & \dots & \dots & \dots & 65 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
d_1^{M_1} & 0.1 & \dots & \dots & \dots & 6000 \\
d_2^1 & 0 & \dots & \dots & \dots & 300 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
d_N^{M_*} & \dots & \dots & \dots & \omega_N^{M_*} & TTL_N^{M_*} \\
\text{sum} & 1 & 1 & \dots & 1 &
\end{array}
$$

Figure 3.2: DNS footprints matrix example

sponding to d_1 column of Figure 3.2. As it can be observed, when accessing to website d_1 the root domain (I in the figure and also in Figure 1.2) presents a weight of 0.35 and a TTL of 178 seconds. At this moment the accumulated weight is 0.35. After resolving the root domain, a subdomain (II in the figure) is resolved. Such subdomain presents a weight of 0.15 and TTL of 60 seconds. At this moment, the accumulated weight is 0.50. Next, a second subdomain (III in the figure) is resolved with a weight of 0.2 and a TTL of 65. After such resolution the accumulated weight is 0.7 which surpasses the predefined α and d_1 is marked as visited. The website is marked as visited until the TTLs of subdomains II and III expires. If website d_1 is visited again, subdomains II and III are resolved again as their corresponding entries are no longer cached and the accumulated weight will surpass again the α marking a second visit for website d_1.

The previous examples have shown how a single website is monitored, in production our proposal monitors several websites in parallel. Figure 3.4 shows some time series S_i for a sample user browsing session. Each line represents the sum of the weights w_i^j of the cached domains for each monitored website during the session.

Note that each monitored website present two different regions in the figure. Whenever a domain is visited, the sum of the weights starts increasing. But after a given time, namely the TTL values, the sum drops as the DNS cache entries start expiring. As it can be observed, whenever a user actually requests a website, several footprints' weights sum increase, but only the one corresponding to the root domain site peaks above threshold α.

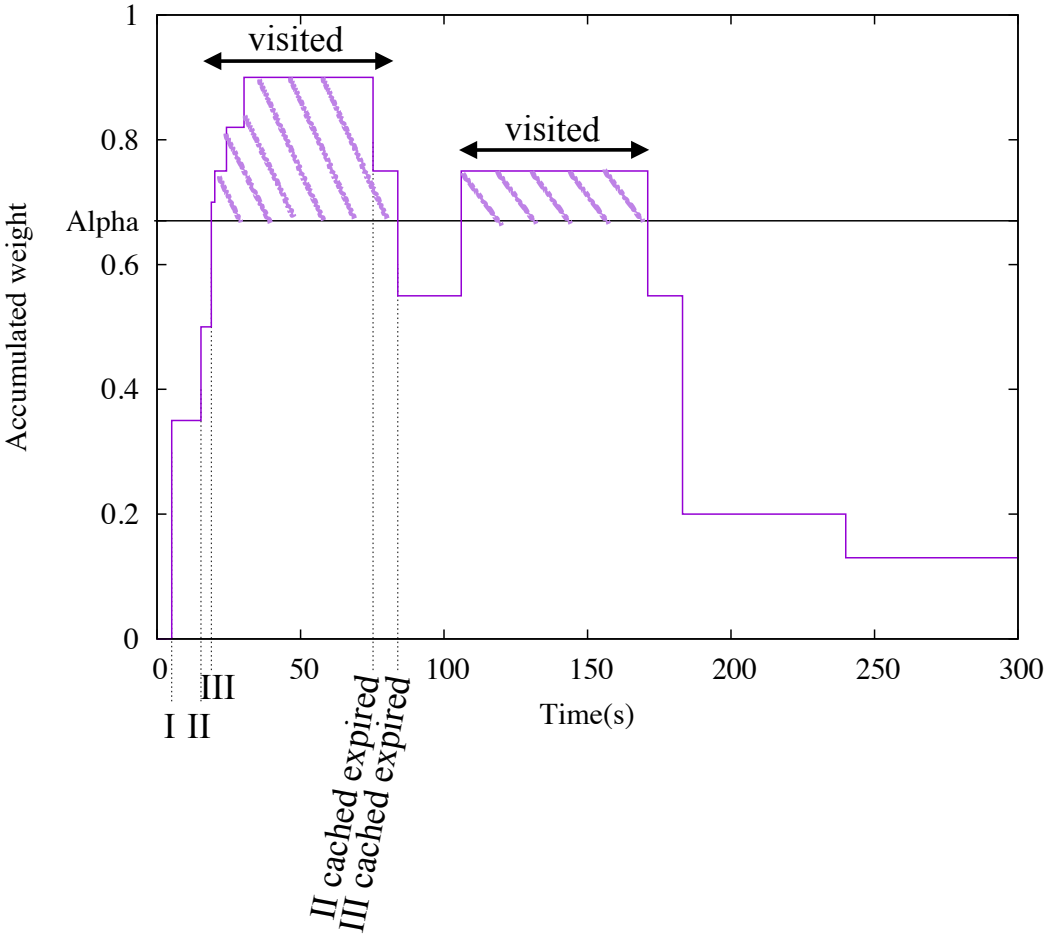

Figure 3.3: Temporal evolution in the detection of visits to the website d_1

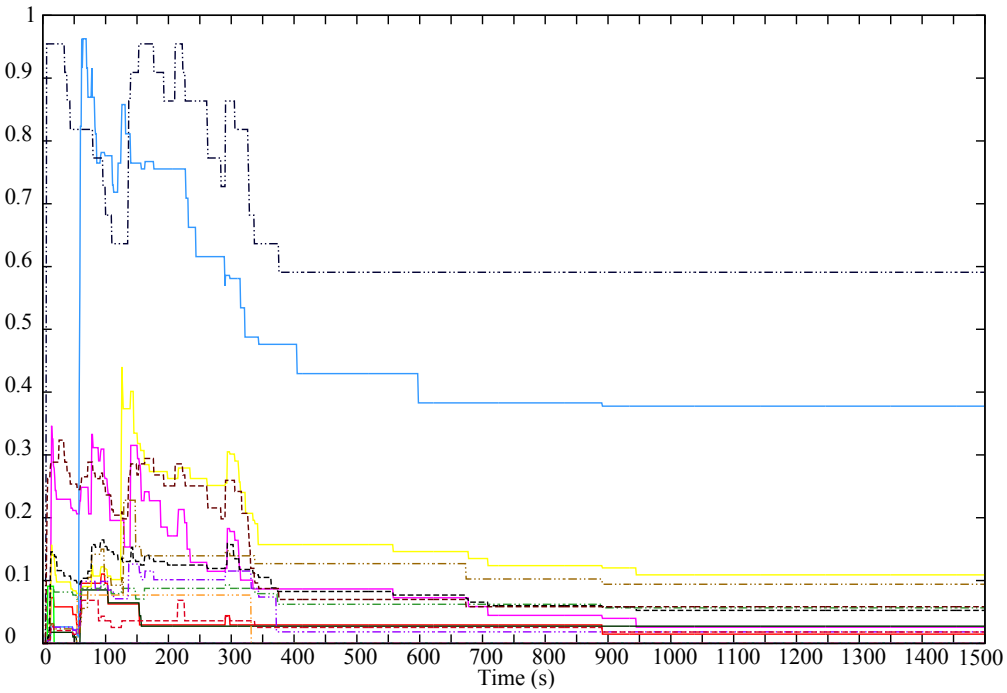

Figure 3.4: Different footprints likelihood during a web browsing session

4

Architecture and implementation

Once the method for detecting website accesses has been described, let us focus on the specific architecture and implementation. Our solution only requires access to the DNS traffic and has been designed to be co-located with the DNS server itself or in a traffic probe that captures the DNS traffic by means of a Switched Port Analyzer (SPAN).

The architecture can be divided into two main modules, the DNS footprint generation and the online web-profile monitoring, which are independent and can run on different computers.

4.1 Footprint generation

The footprint generation module's goal is to abstract the process of calculating the footprints and keep an updated set of footprints that will be used by the online monitoring module.

An important aspect to consider is that self-triggered resolutions for a given domain query are not always the same, and vary with the time, operating system and web browser. To mitigate this effect, each footprint F_i and the corresponding weights w_i^j have to be refined to reflect the different behavior that potentially each computer can present.

This is achieved by defining partial and final footprints. Partial footprints are obtained directly by querying the monitoring websites, and are afterwards refined to build a final footprint for each monitored website d_i, which will be used by the monitoring module. The footprint generation process is illustrated in Figure 4.1.

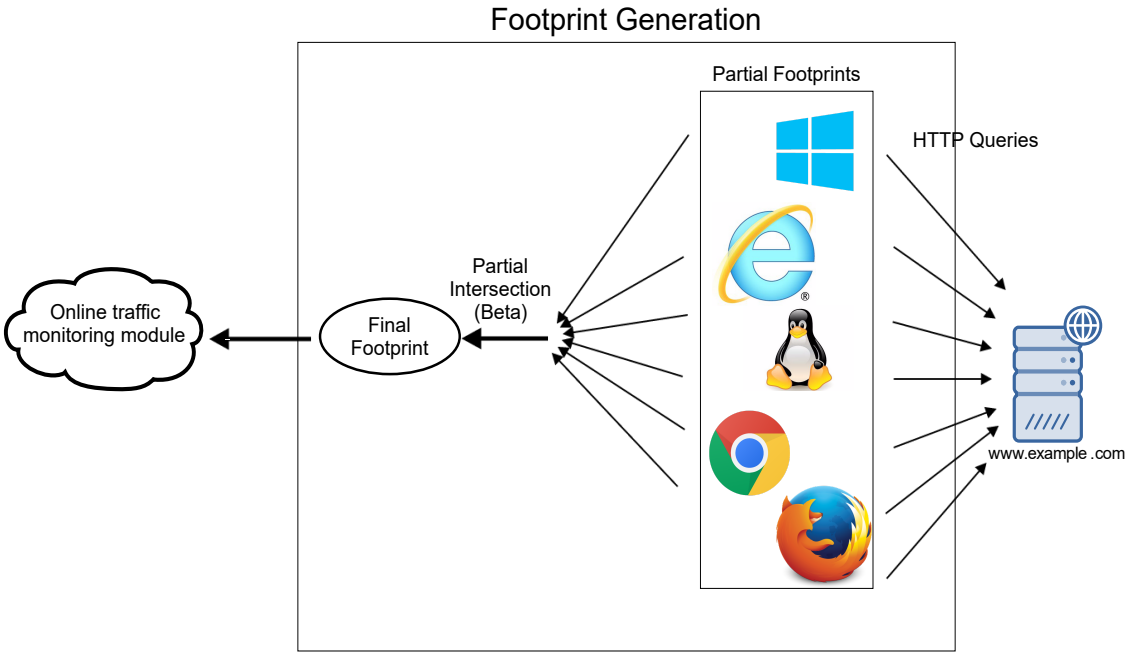

Figure 4.1: Partial and final footprints generation process.

4.1.1 Partial footprints construction

As websites quickly evolve and update their content, it is necessary to keep an updated footprint for each of them, refreshing them periodically.

Outside the monitoring platform, an HTTP software robot periodically polls the monitored domains using different browsers (Mozilla Firefox and Google Chrome). Every time a website is queried, all traffic generated is captured and filtered, keeping only DNS resolutions and elaborating a list of resolved domains, which is stored as a partial footprint of the domain.

To allow partial footprints generation on a server, which usually does not have a desktop to display the website in the browser, the virtual framebuffer *xvfb* is used. This allows to run a web browser on a server without having an actual display.

However, it is necessary to wait for some time between each query to make sure that all self-triggered domains are completely resolved before starting to update the next footprint. As an example, we follow a 6-hours footprint update policy, using a 60-second interval for each request to ensure that all the self-triggered domains are resolved in time and are added to the footprint.

It is important to note that most of the websites present a completely different layout or even a different domain name when accessed from a mobile device. Including such cases in a footprint would cause footprints to have a completely different structure and would reduce the

accuracy of the method. Therefore, mobile devices should be studied separately and different and independent footprints should be generated if mobile traffic wants to be analyzed.

4.1.2 Final footprints construction

Each set of partial footprints for a website, which is periodically updated, has to be combined into a single final footprint that will be used by the monitoring module.

To achieve this, a partial intersection is performed. That is, having n partial footprints of a website, a domain is added to the final footprint if it appears in at least $n \cdot \beta$ of them.

The β parameter should be tuned to maximize the robustness of the footprint. Having $\beta \sim 0$ would turn the partial intersection into a union, and the final footprint would not be resilient to minor changes or variations in the website. On the other hand, having $\beta \sim 1$ would turn the partial intersection into a strict intersection, greatly shrinking footprints size, and the footprint would become empty if the website is down or network connection is lost once.

After all the final footprints have been constructed, each being a list of domain names, the weights w_i^j for each subdomain are obtained as explained in Chapter 3 and the final footprints are delivered to the monitoring module.

4.2 Online web-profile monitoring

Once the footprints have been constructed, we turn our attention to how apply them to identify website accesses. As DNSprints is designed for running in a DNS server that serves millions of users, special performance requirements have to be considered. To maximize performance, DNSprints has been implemented entirely in C language and is able to analyze traffic online directly from a network interface card (NIC). The implementation also avoids linear or more complex algorithms by using hash tables to store most of the data.

This online monitoring module only processes DNS answers, as they are the ones which contain the corresponding DNS cache TTL. When a DNS answer is received, it must first be parsed to identify the domains it is resolving, as a single packet can contain several DNS answers. It is also important to consider that the DNS protocol compresses domain names to reduce traffic throughput, so names must me uncompressed on the fly [21]. Once the domain names are identified, the ones which do not belong to any footprint are ignored.

Each time a domain resolution is observed the algorithm analyzes it in an event-driven way. Note that for every domain name there can be several different footprints that contain such domain name, presenting a different weight in each of them. It is also necessary to remove DNS cache entries as they expire. To achieve this, a heap-based priority queue keeps track of all cache entries, ordering them from most recent to latest expiration time.

To process DNS answers on the fly, we propose Algorithm 1 that associates each domain name with all the footprints it belongs and adds the corresponding weight in each of them. Every time a DNS query for a domain name d_i^j that belongs to F_i is received, S_i and the priority queue are modified as follows:

- If d_i^j is not cached when the query is received (e.g on the first visit to that domain), the current value for S_i is increased by w_i^j, which is the corresponding weight associated to such subdomain name, and d_i^j is added to the table of cached domains.
- If d_i^j is already cached by the user, the weight (w_i^j) decrease event is delayed until time T if it is larger than the current one.

In addition, whenever a domain is resolved and cached, an event is added to the priority queue to indicate that the domain's weights should be subtracted from S_i after the time T, indicated by the DNS server, has elapsed. After the time T has elapsed an event handler triggers and the event is automatically pulled from the queue, removing the DNS cache entry and subtracting the corresponding weights. Algorithm 2 illustrates this.

This allows the algorithm to compute the time series S_i for each user on the fly, leveraging the event-driven priority queue to store the DNS cache expiration time of every cache entry. As the priority queue will automatically subtract the corresponding weight whenever a cache entry is expired, it is only necessary to store the current value of S_i. A new visit is considered each time S_i surpasses α value and the end of the visit is considered when S_i drops below α.

The key restriction for DNSprints is that it needs to replicate the DNS cache for each user and update it in real time. Inevitably, this is very memory demanding. In order to allow a fast access to the cache of any user the cache is stored in a custom implementation of a hash table that minimizes dynamic reallocation and therefore provides fast insertion, reading, and deletion.

As DNSprints is very memory demanding, it is necessary to optimize memory consumption and data structures used. To avoid storing the same domain string in memory twice and to allow fast indexing using domains, DNSprints assigns an index to each website name or domain name. A hash table stores this name-index association, allowing fast access in both directions (translating name to index and vice versa).

Algorithm 1 DNSprints answers processing algorithm

for all domain D in request **do**
  if D was not cached **then**
    Add D to cached domains table
    Add new cache expiration event to the priority queue.
    for all F_i such as $D \in F_i$ **do**
      Add corresponding weight w_i^j to S_i
      if $S_i \geq \alpha$ and $S_i - w_i^j < \alpha$ **then**
        Report started visit to website d_i
      end if
    end for
  else
    Update cache expiration time for D
  end if
end for

Algorithm 2 DNSprints priority queue algorithm

for all expired cache domain C **do**
  Remove C from cached domains table
  for all F_i such as $C \in F_i$ **do**
    Subtract corresponding weight w_i^j from S_i
    if $S_i < \alpha$ and $S_i + w_i^j \geq \alpha$ **then**
      Report finished visit to website d_i
    end if
  end for
end for

5

Datasets, Results and Discussion

5.1 DNS Footprints analysis

Once the footprints of each website are constructed, we turn our attention to their structure. This section will describe features observed while generating footprints from different websites using DNSprints:

- Footprints size is very dependent from the website and the browser used: there is no general pattern that footprints fall in. Depending on the website and browser the number of domains in a footprint span from less than 10 to more than 400.
- TTLs of websites' root domains fall in a wide range: As with the footprint size, the TTL of the root domain name greatly varies depending on the websites being monitored, ranging from a few seconds to several hours.
- Consecutively querying a website twice with the same device and browser (with no DNS caching) consecutively does not generate the same footprint for both queries.

5.2 DNS online monitoring analysis

Once the implementation and the method for detecting visits to tangled websites using DNS traffic have been described, the experimental evaluation is in order. Our objective is evaluating the method in terms of accuracy, reliability and performance.

5.2.1 Accuracy evaluation

Experimental datasets

To evaluate the method we have focused on the application over a set of users that can be precisely monitored, capturing all the DNS traffic for evaluation and all the web traffic. To do so, we have captured DNS and web traffic of a research laboratory of our university. With this traffic, we have generated a dataset that contains records with the visited websites along with their timestamp for several users, i.e, the ground truth. The captured traffic has been generated by desktop and laptop computers with both diverse operating systems and browsers (including those used to construct the footprints). Additionally, to test the method in a realistic environment the set D of monitored websites has been chosen as the union of the following subsets:

1. The top 50 news websites wordlwide [22]
2. The top 50 business websites worldwide [23]
3. The most visited nationwide business and news websites.

News and business websites have been chosen since they are likely to benefit from the user-trends and behavior analysis. Moreover their root domains present large DNS cache TTL. Since several websites in the news and business subsets are popular in the US and most of them are not frequently visited by our test users, they have been encouraged to add to their regular browsing some randomly-picked websites from such lists to provide a more exhaustive testing. Furthermore, with the addition of the subset of local business and news websites we cover the typical behavior (most visited local websites) of such test users. The dataset contains traffic corresponding to the monitored websites D and traffic to other non-monitored websites which helps when measuring noise tolerance.

In more detail, to build the ground truth of the visited websites, we have considered that a visit to a website d_i has taken place when a user requests by means of an HTTP GET message either the "/" or "/index.*" resource [24]. Since only queries performed by a user using a web browser, and no other queries automatically generated by banners, ads or built-in content, should be considered as visits, only GET requests to these default resources are considered for generating the ground truth of visits. As a single website request can open several HTTP connections to the same domain, connections to a same domain separated less than 15 seconds [12] are merged into a single one to avoid duplicates. Due the existing delay between the HTTP request and its corresponding DNS traffic, HTTP connection lengths have been extended in a 15 seconds window surrounding the GET request timestamp. Connections using HTTPS are also considered as when a user tries to access the "/" or "index.*" resource of a domain for the first time, the browser generates an HTTP GET request by default as it does not know beforehand if such website accepts HTTPS.

Finally, the obtained ground truth set, $H = \{h_i = (u_i, d_i, t_{0i}, t_{1i})\}$, $i = 1, 2, \ldots M_H$, is conformed by tuples that contain the user (u_i), domain (d_i), initial time (t_{0i}) and the finishing time (t_{1i}).

Tests and results

To evaluate the accuracy of the method, the DNS traffic from the experimental dataset has been evaluated, generating a set of estimated visits G analogous to the previously defined one, $G = \{g_i = (u_i, d_i, t_{0i}, t_{1i})\}$, $i = 1, 2, \ldots M_G$. After this, two sets of visits, H corresponding to our ground truth HTTP analysis method, and G corresponding to our DNS-based test method are presented.

Even though this scenario is not strictly a binary classifier case, to evaluate the accuracy of the method we rely on the True Positive Rate (TPR) and the False Positive Rate (FPR). Using this data, we can build a ROC-like curve for different values of the parameter α so that we can illustrate the accuracy of the method and help choosing the optimal value for it.

Intuitively, the TPR can be defined as the ratio between the number of visits that are present in the set G as well as in the set H and the total number of visits present in H. Similarly, the FPR can be defined as the ratio between the number of visits that are present in the set G but are not present in the set H and the total number of visits of the set G. Formally, such magnitudes are defined as follows:

$$TPR = \frac{|\{h_i \in H : \exists g_j \in G, u_i = u_j, d_i = d_j, [t_{0i}, t_{1i}] \cap [t_{0j}, t_{1j}] \neq \emptyset\}|}{|M_H|}$$

$$FPR = 1 - \frac{|\{g_i \in G : \exists h_j \in H, u_i = u_j, d_i = d_j, [t_{0i}, t_{1i}] \cap [t_{0j}, t_{1j}] \neq \emptyset\}|}{|M_G|}$$

Figure 5.1 shows the ROC-like curve for 100 different values of α parameter. Note that the points located in the upper left corner correspond with the best values being the ideal classifier the one that presents TPR=1 and FPR=0. Based on the analysis of the obtained rates of the curve we have chosen $\alpha = 0.65$. With this optimal value for α, 88% TPR and 7% FPR are achieved. As the accuracy test can be run automatically, it is possible to tune the value of α on the fly by constantly monitoring a representative subset of users and updating the test results dynamically by adapting the α value to reduce the distance to the point (0,1).

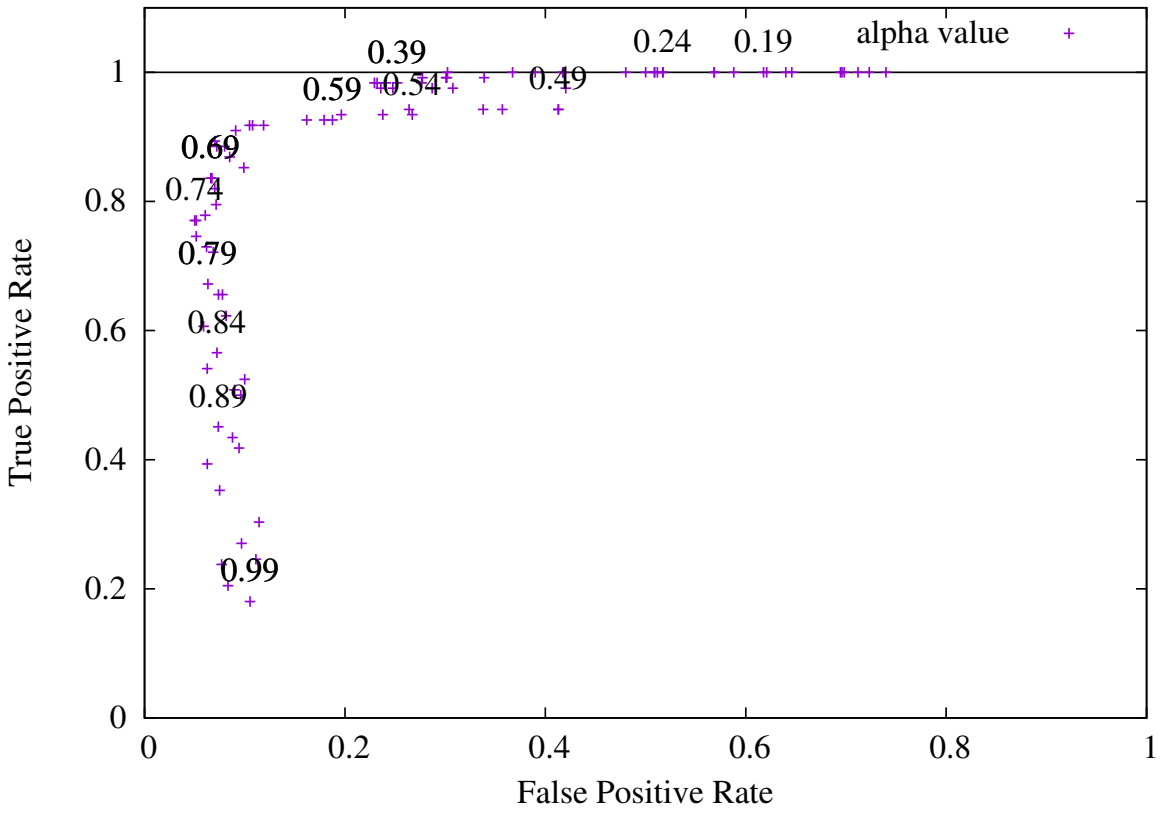

Figure 5.1: ROC-like curve showing FPR vs TPR for different values of α

5.2.2 Performance evaluation

As DNSprints is designed to run in high-throughput environments we have evaluated its performance on a worst-case scenario. Our objective is to evaluate the performance of the online monitoring module, as footprint generation can be performed offline and does not have strong requirements.

Throughput rate

To evaluate DNSprints' throughput rate, we have used an HTTP robot to randomly query monitored websites periodically during a week, capturing DNS traffic into a file and afterwards running DNSprints on that file. The set of monitored websites consists of more than 200 websites taken from the top websites of different categories in Spain, focusing especially on news and business sites, as those are the main candidates to exploit DNSprints' features. The footprints for all monitored websites contain more than 5000 different domain names in total.

This is a worst-case scenario since all DNS queries and responses analyzed will correspond to monitored websites. In an unbiased production environment, a large portion of DNS queries and responses will correspond to non-monitored websites, and will be ignored by DNSprints, thus increasing its performance.

Running DNSprints over the DNS traffic file generated, we have obtained a performance of 500,000 DNS packets per second. This test has been performed on a computer that contained 16 GB of DDR3 memory, and a 4 core Intel i5-3570K processor running at 3.40 GHz on an ASUS P8H77-M LE motherboard.

Memory requirements

The greatest disadvantage of DNSprints is its high memory consumption when run over a large set of websites and users. As it was not possible to access DNS traffic from a functioning DNS server that serves thousand or millions of users, a theoretical analysis of memory consumption is provided.

DNSprints' memory consumption depends on three variables: the number of websites being monitored, the number of domain names in those websites' footprints and the number of users. As all data structures grow at most linearly with any of these variables, structures which size depends on at most one of those variables are ignored for this section's study, since their memory consumption is several orders of magnitude lower than memory consumption of those which depend on two or more variables.

DNSprints has the following relevant structures for memory consumption:

- Footprints: DNSprints stores the DNS footprint of each monitored website. The total number of entries in footprints is approximately equal to ($monitored\ websites \times average\ footprint\ size$). It will be slightly lower as domains that are present in two or more footprints are stored just once. Each footprint entry's size is equal to the domain name's length.
- Cache table and priority queue: as is was previously introduced, DNSprints needs to store the DNS cache data for every monitored website and user. This has been implemented using a hash table and a priority queue, which size grow linearly with the total number of cached domains by each user at a given time. Even though its size depends on both number of users and number of domains in footprints, it is very dependant on user's activity and the popularity of the monitored domains among users. Having very active users and very popular websites would lead to storing large amounts of cache entries and high memory consumption. Each cache entry stored requires approximately 40 Bytes.
- S_i weight counters: for each user and monitored website, the current state of S_i is stored, that is, an 8 Bytes floating point number. As this is expected to be the largest data structure on a high-performance environment analyzing millions of users, its size could be reduced by reducing S_i precision to a 4 Bytes floating point value.

6

Conclusions and future work

We have presented a method for identifying accesses to websites using DNS footprinting of interest for web analytics. The method exploits the fact that current websites contain a set of contents such as ads, banners, other services of the owner of the website and social networks among others. Such contents must be downloaded in order to fully load the website that the user introduced in the address bar, in other words they are self-triggered. Our proposal takes advantage of such entanglement on websites to create a DNS footprint that allows identifying an access to a website by expecting not only DNS answers for such website but also answers for the self-triggered contents.

The proposed method has been implemented in C language in a tool called DNSprints that has been evaluated in terms of accuracy using real traces from a research laboratory with satisfactory results, that is obtaining a false positive rate of 7% and a true positive rate of 88%. In addition, performance tests have been run, obtaining rates of 500,000 packets per second.

As future work we plan to extend the method to focus on mobile web traffic that today is an important market both for ISPs and Internet enterprises. Additionally, the impact that ads or social media blockers such as Adblock or Ghostery browser addons may have in the generation of footprints must be studied. Although such resources tend to have low TTLs and, given that they are very common in the set of websites under study, their relevance in footprints should be low. In Section 5.2.1 we have pointed to the possibility of dynamically calculating the value of the parameter α. Finally we turn our interest in studying how DNS prefetching (many modern browsers resolve domain names before users actually request them to reduce latency) affects the DNS footprint and how DNSprints could exploit it.

There are as well different applications of the DNS footprint, and DNSprints could be extended in different ways that are not covered and studied in this document. As an example, DNSprints could not only be used to identify visits to websites, but also to different platforms such as applications that connect to the Internet, studying the domains that those applications resolve and creating a footprint for them. This is especially relevant as a large portion of accesses to the Internet from mobile devices is done from a specific application and not from a web browser.

Another extension for DNSprints could be to track not only which websites are being accessed, but also the characteristics of the devices they are being accessed with. As we have stated in Section 5.1, the DNS footprint varies depending on the device, OS and browser used. Therefore, keeping DNS footprints for different browsers or devices could reveal not only that a visit took place, but also provide detailed insight on how the visit was performed.

To conclude, we plan to implant DNSprints in a production DNS server of a large ISP that serves millions of users, as an attempt to find DNSprints' boundaries and further evaluate its performance, accuracy and applicability.

Glossary of acronyms

- **DNS**: Domain Name System
- **IP**: Internet Protocol
- **ISP**: Internet Service Provider
- **HTTP**: Hypertext Transfer Protocol
- **HTTPS**: Hypertext Transfer Protocol Secure
- **TTL**: Time To Live
- **OS**: Operating System
- **UDP**: User Datagram Protocol
- **DPI**: Deep Packet Inspection
- **SPAN**: Switched Port Analyzer
- **NIC**: Network Interface Card
- **US**: United States
- **TPR**: True Positive Rate
- **FPR**: False Positive Rate
- **ROC**: Receiver Operating Characteristic

Bibliography

[1] "Dynamic profiling of consumers for customized offerings over the Internet: a model and analysis," *Decision Support Systems*, vol. 32, no. 2, pp. 117–134, 2001.

[2] "Web user behavioral profiling for user identification," *Decision Support Systems*, vol. 49, no. 3, pp. 261–271, 2010.

[3] "Google adwords." [Online]. Available: https://www.google.com/adwords

[4] M. Lee, "Google ads and the blindspot debate," *Media, Culture & Society*, vol. 33, no. 3, pp. 433–447, 2011.

[5] "About Alexa data." [Online]. Available: http://www.alexa.com/about

[6] A. Juels, M. Jakobsson, and T. Jagatic, "Cache cookies for browser authentication," in *IEEE Symposium on Security and Privacy*, 2006, pp. 300–305.

[7] J. L. Garcia-Dorado, A. Finamore, M. Mellia, M. Meo, and M. Munafo, "Characterization of ISP traffic: Trends, user habits, and access technology impact," *IEEE Transactions on Network and Service Management*, vol. 9, no. 2, pp. 142–155, 2012.

[8] D. Plonka and P. Barford, "Context-aware clustering of DNS query traffic," in *ACM SIGCOMM Conference on Internet Measurement*, 2008, pp. 217–230.

[9] ——, "Flexible traffic and host profiling via DNS rendezvous," in *Workshop on Securing and Trusting Internet Names*, 2011, pp. 29–36.

[10] I. N. Bermudez, M. Mellia, M. M. Munafo, R. Keralapura, and A. Nucci, "DNS to the rescue: Discerning content and services in a tangled web," in *ACM SIGCOMM Conference on Internet Measurement*, 2012, pp. 413–426.

[11] T. Mori, T. Inoue, A. Shimoda, K. Sato, K. Ishibashi, and S. Goto, "SFMap: Inferring services over encrypted web flows using dynamical domain name graphs," in *Workshop on Traffic Monitoring and Analysis*, 2015, pp. 126–139.

[12] M. Fullmer and S. Romig, "The OSU flowtools package and Cisco Netflow logs," in *USENIX LISA Conference*, 2000, pp. 291–304.

[13] H. Choi and H. Lee, "Identifying botnets by capturing group activities in DNS traffic," *Computer Networks*, vol. 56, no. 1, pp. 20–33, 2012.

[14] "Comscore inc." [Online]. Available: http://www.comscore.com

[15] "Netratings inc." [Online]. Available: http://www.nielsen-netrating.com

[16] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband Internet traffic," in *ACM SIGCOMM Conference on Internet Measurement*, 2009, pp. 90–102.

[17] "The measurement factory, DNS tools." [Online]. Available: http://dns.measurement-factory.com/tools/

[18] D. Herrmann, C. Banse, and H. Federrath, "Behavior-based tracking: Exploiting characteristic patterns in DNS traffic," *Computers & Security*, vol. 39, Part A, pp. 17–33, 2013.

[19] M. Rajab, F. Monrose, A. Terzis, and N. Provos, "Peeking through the cloud: DNS-based estimation and its applications," in *Applied Cryptography and Network Security*, 2008, pp. 21–38.

[20] T. Callahan, M. Allman, and M. Rabinovich, "On modern dns behavior and properties," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 7–15, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2500098.2500100

[21] P. Mockapetris, "RFC 1035: Domain Names - implementation and specification," 1987.

[22] "Alexa top 50 news websites." [Online]. Available: http://www.alexa.com/topsites/category/Top/News

[23] "Alexa top 50 business websites." [Online]. Available: http://www.alexa.com/topsites/category/Top/Business

[24] J. Reschke and R. Fielding, "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and content," 2014.